PCT/IB03/02842
20.06.03

Europäisches Patentamt | European Patent Office | Office européen des brevets

Rec'd PCT/PTO 21 DEC 2004

REC'D 24 JUL 2003
WIPO PCT

Bescheinigung Certificate Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

Patentanmeldung Nr. Patent application No. Demande de brevet n°

02291623.3

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

[illegible]

R C van Dijk

Europäisches Patentamt

European Patent Office

Office européen des brevets

Anmeldung Nr:
Application no.: 02291623.3
Demande no:

Anmeldetag:
Date of filing: 28.06.02
Date de dépôt:

Anmelder/Applicant(s)/Demandeur(s):

Koninklijke Philips Electronics N.V.
Groenewoudseweg 1
5621 BA Eindhoven
PAYS-BAS

Bezeichnung der Erfindung/Title of the invention/Titre de l'invention:
(Falls die Bezeichnung der Erfindung nicht angegeben ist, siehe Beschreibung.
If no title is shown please refer to the description.
Si aucun titre n'est indiqué se referer à la description.)

Software download into a receiver

In Anspruch genommene Prioriät(en) / Priority(ies) claimed /Priorité(s) revendiquée(s)
Staat/Tag/Aktenzeichen/State/Date/File no./Pays/Date/Numéro de dépôt:

Internationale Patentklassifikation/International Patent Classification/
Classification internationale des brevets:

G06F9/40

Am Anmeldetag benannte Vertragstaaten/Contracting states designated at date of filing/Etats contractants désignées lors du dépôt:

AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR

SOFTWARE DOWNLOAD INTO A RECEIVER

FIELD OF THE INVENTION

The invention relates to transmission systems. More particularly, it relates to a method of downloading software programs into a storage unit, in a transmission system.

The invention also relates to a transmission system and to a receiver of said transmission system.

It also relates to a computer program product for carrying out the method mentioned above and to a signal for carrying the computer program.

The invention applies to any communication system, including mobile communication systems. It particularly applies to broadcasting systems such as the DSS (Digital Satellite System) system and the DVB (Digital Video Broadcasting) system.

BACKGROUND ART

Bootstrap Loaders (BSLs) are widely spread in consumer products. They manage the download of software applications and allow to easily upgrading the products by replacing the current software application with a new software application. For safety reasons, BSLs are generally stored in a protected memory area where the software boots after hardware reset. The memory area is protected electrically in factory and cannot be unprotected without hardware intervention. Those BSLs are generally dedicated to a technology, including e.g. specific transport protocols, which are not standardized, linked to the system used and to a kind of application that has to respect the same protocol. When changing technology, in order to download a new software application, it is thus necessary to also download the associated new BSL. The current BSL will thus not be used any more, although it uses protected memory space, which is now wasted for the rest of the application. On the other hand, the new BSL is generally downloaded in a memory area, which is not protected and can thus suffer from damages caused e.g. by voluntary or involuntary software erasure.

SUMMARY OF THE INVENTION

It is an object of the invention to provide a method of downloading software programs into a receiver, which avoids the drawbacks mentioned above and thus yields a better quality of service for the end user. The method enables replacing complete embedded software program including a boot code (or Bootstrap Loader) and application code (or main application) dedicated to a technology (e.g. the DSS technology) by another one dedicated to another technology (e.g. the DVB technology). The change in technology mentioned above may refer to a transmission technology, such as DVB or DSS, but can also refer to a conditional access technology and / or broadcaster technology, or to any non-compatible change in the global system.

To this end, the invention proposes a first method, such as mentioned in the opening paragraph, wherein the software programs include a boot code and an application code, the boot code allowing downloading the application code, the storage unit comprising at least a current software program stored including a current boot code, the method comprising:

- upon a download request, downloading a new boot code in a location, which does not overwrites the current boot code,
- indicating that the new boot code replaces the current boot code,
- downloading a new application code associated to the new boot code in a location, which does not overwrites the new boot code,,
- indicating that the new application code is valid.

This first method advantageously allows re-using the memory space used by the current boot code, while avoiding that the current boot code can be damaged before the new boot code is validly downloaded.

In accordance with another embodiment of the invention, a second method is proposed, comprising:

- defining a boot sector for jumping on a position of the storage unit where a boot code is stored in order to validate the use of said boot code, the boot sector initially pointing on the first position, where the current boot code is stored
- upon a download request, downloading a new software program in a second position including a new boot code and a new application code,
- jumping on the second position where the new boot code is stored.

In the second method, the operations can advantageously be done in a secure, memory safe way including ability to go back to the previous software, the manufacturer keeping the possibility to ignore the second technology. Moreover, the memory space used for the current boot code can advantageously be re-used for storing e.g. the new application code, which is efficient.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention and additional features, which may be optionally used to implement the invention to advantage, are apparent from and will be elucidated with reference to the drawings described hereinafter, wherein:

- Fig. 1 is a schematics for illustrating an example of a first method in accordance with a first embodiment of the invention,
- Fig. 2 is a schematics for illustrating an example of a second method in accordance with a second embodiment of the invention,
- Fig. 3 is a schematics for illustrating an example of a system in accordance with the invention.

DETAILED DESCRIPTION OF THE DRAWINGS

It will be described a technique to do efficient and reliable downloading of new software into a consumer equipment device, called a receiver, like e.g. a set-top box or a digital TV. Efficiency is estimated in terms of non-volatile memory needed to do the download; reliability is estimated in terms of the resistance of the process against interruption. The download can be done indifferently e.g. over the air, by cable transmissions, satellite links, using local download or via hardware module, such as e.g. cards, external devices, etc.).

The new software needs to be written into persistent memory (e.g. a FLASH memory). If the process is interrupted (e.g. by a power switch off), whereas the process of overwriting the old software with the new one is not completely finished, the software will generally no longer work. One solution is to avoid overwriting the old software. But then the persistent memory used to store the "old software" is definitively wasted, which is not efficient.

To solve the problem, two methods are proposed for downloading software programs into a storage unit. In accordance with these methods, the software programs include a boot code and an application code, the boot code allowing downloading the program code. The storage unit comprises at least a current software program stored including a current boot code and a current application code.

The first method, illustrated in Fig. 1, comprises:

- upon a download request, downloading a new boot code in a location, which does not overwrites the current boot code,
- indicating that the new boot code replaces the current boot code,
- indicating that the current application code may be corrupted,
- downloading a new application code associated to the new boot code in a location,

which does not overwrites the new boot code,
- indicating that the new application code replaces the current application code.

Fig. 1 illustrates in greater details this first method with a 5-step process. The blocks represent memory areas of the storage unit.

- Start: the current software is split into two parts: one which is able to perform a software download, called the current boot code and denoted CBC, and the other part, which complements this functionality to make the total application, called the current application code, denoted CAC. Persistent flags, called "indicators" are used and combine the function of flagging reliability or not with the position where the respective code can be found in persistent memory. A boot code indicator flag, denoted BCI, indicates which boot code is valid to be used. An application code indicator flag, denoted ACI, indicates which application code is valid to be used.
- Step 1: Before new software is downloaded, the flag ACI is set to signal that the old / current application code may be corrupted.
- Step 2: The new boot code NBC is written into persistent storage memory. In the process the old / current application code CAC may be overwritten.
- Step 3: After successfully writing the new code, the flag BCI is set to indicate that from now on the new boot code NBC should be used rather than the old / current one CBC.
- Step 4: The new application code NAC can then be written in persistent memory.
- Step 5: When the new application code is successfully written in persistent memory, a persistent flag is set to indicate the new application code is ready.

In steps 2, 3 and 4, any operation of writing into the persistent memory may eventually be preceded by a provisional load into a volatile memory e.g. a RAM (Random Access Memory) in order that the integrity of the data can be checked.

As a security measure, the old / current boot-code can be copied (backed up) to another part of the persistent memory of the receiver before the new boot code is written over the old / current boot code, or the old / current boot-code can be copied to an alternative location while writing the new boot-code into the memory previously storing the old / current boot code, such that if the process of copying the new code is interrupted, the receiver can detect this situation and restore the old / current boot code.

For ultimate reliability in step 3 it is possible to use a single memory bit to indicate which of the new boot images is correct. In case the process of writing this bit is interrupted, it may read as 0 or 1, but the outcome is arbitrary: any of the two images is good.

It is also possible that the application code can be split up in multiple parts. If only the boot code needs to be replaced it may be necessary to overwrite parts of the application code with the new boot code. Then only the overwritten parts of the application code need to be re-loaded.

As far as implementation is concerned, a "flash" file system can advantageously be used for carrying out the method mentioned above, for the software management. In that case a manager application has to ensure that it first makes sufficient place in the file system to allow a new boot application to be stored before the old one is overwritten, and that the flagging/indication is done reliably, such that there cannot occur a inconsistent state if the process is stopped half way. In particular it needs to be guaranteed that the file pointers can be updated "atomically": i.e. in one action, or that the single bit approach is used. It is also possible to check if the boot-code is valid by checking a checksum computed over the whole boot-code. If the image is partially overwritten it will turn out to be invalid. Then the system can look for an alternative.

Last but not least, in some systems it may be important or convenient that the boot-code is located in the same position in memory since there may be constraints on its location. In such systems, instead of writing the new boot code in another location, the old boot code can be moved there before or during the copying of the new boot code. In case the process is interrupted, the device can then still recover the old boot code by copying back the new code. Detection that the (partially written) boot-code is not complete can be done through a "pointer", a "bit" or a "checksum".

The major problem, which may be raised when replacing the current boot code, is that it is stored on the boot sector and consequently, an inopportune power off during writing would definitely crash the box. An alternative solution to avoid this, which consists in the second method according to the invention, is that the software would boot on another sector that would never change and that would just take the decision to jump onto addresses where there is always a boot code stored, preferably in a protected memory area. Only two distinct addresses address1 and address2 are indicated in Fig. 2 but there may be more addresses. This is for safety reason: if an inopportune power off occurs while writing, there will always be a valid BSL to be executed. This boot sector does not need many software resources. It is a very little application that does not overload the global size so much. As it will never be erased, it can even be stored in ROM memory. In case of weak hardware constraints, the boot sector can include a proprietary local download system, e.g. though serial link.

This second method of downloading software programs into the storage unit of a receiver before a change in the system occurs is described below. It comprises:

- defining a boot sector for jumping on a position of the storage unit where a boot code is stored in order to validate the use of said boot code, the boot sector initially pointing on the first position, where the current boot code is stored
- upon a download request due to a change in the system, downloading a new software program in a second position including a new boot code and a new application code, which is meant to work in the new system,
- jumping on the second position where the new boot code is stored.

The memory space used for the current boot code can advantageously be re-used e.g. for storing the new application code.

Eventually, in systems where it is important that the boot code is located always at the same position, the above mentioned method is completed with the following steps of:
- replacing the current boot code with the new boot code at the first position,
- jumping on the first position.

As mentioned above, the boot sector is preferably located in a protected storage area, which can be inside the storage unit or separate from the storage unit.

In a preferred embodiment of the second method, the boot code and/or the application code are stored in an area of the storage unit, which area can be alternatively protected and unprotected to be overwritten under specific software conditions. This preferred embodiment takes advantage of a new technology of persistent memory that enables to protect/unprotect memory area by software instructions.

The new software program may include an intermediate application code, which is a link between the current application code and the new application code enabling a user to parameterize his receiver into the new system, e.g. for changing a subscriber card, an antennae, an antennae pointing, for calling a phone center to validate new services, etc.

Letting aside implementation aspects, a preferred embodiment of the second method can be summarized as follows:
- Step 1: when a download request is detected; the current boot code BSL1 downloads normally the new application including the new boot code BSL2.
- Step 2: the new boot code BSL2 memory area is protected.
- Step 3: the current boot code BSL1 memory area is unprotected. From now, the boot sector will jump onto the new boot code BSL2.
- Step 4: the current boot code BSL1 is erased.

- Step 5: normal download to the new boot code BSL2 can be requested.
- End: the new boot code BSL2 loads Appli2 in accordance with the second protocol; returning back to the normal mode; the operation being consequently completely reversible.

Fig. 2 illustrates in greater details an example embodiment of this second method. The blocks represent memory areas of the storage unit. The current and new boot codes are denoted BSL1 and BSL2, respectively. The current and new application codes are denoted Appli1 and Appli2, respectively. The boot sector is denoted Boot. The method comprises the following steps:

-Start: The boot sector normally jumps onto the current boot code BSL1 that will jump onto the current application Appli1 if no download is requested.

- Step 1: Appli1 detects a download request: it resets the box that boots on the current boot code BSL1 that downloads normally the new application, but which is actually a link between the future new boot code BSL2 and an intermediate valid application called Switch Appli. Finally, the current boot code BSL1 jumps onto the intermediate valid application Switch Appli. At this stage, the new boot code BSL2 is simply concatenated to the Switch Appli, as application data, and is not yet functional. Therefore, the current boot code BSL1 jumps directly to Switch Appli entry point. The point in linking the new boot code BSL2 with the Switch Appli is to limit the number of downloads.
- Step 2: Switch Appli protects the memory area where it is stored.
- Step 3: Switch Appli unprotects the memory area where BSL1 is stored. From now on, after a reset, the boot sector will jump onto Switch Appli. Switch Appli can display a man machine interface to warn a user of a change and eventually ask him to adapt some parameters of the system, such as, e.g. change the orientation of an antennae, enter a new access code for accessing another network, etc.
- Step 4: Switch Appli erases the current boot code BSL1 and eventually or if required, writes the new boot code BSL2 to its final location, e.g. in place of the current boot code BSL1
- Step 5: Switch Appli protects the new boot code BSL2 memory area. Consequently, from now on, the boot sector will jump onto the new boot code BSL2.
- Step 6: Switch Appli unprotects the memory area where it is stored; requests a normal download to the new boot code BSL2 and resets the box.
- End: BSL2 loads Appli2 respecting the second protocol, we are back to the normal mode. This operation is consequently completely reversible.

In the example illustrated in Fig. 2, at each reset, the boot sector will jump onto the highest protected sector (see arrows on the figures). But other conditions may be defined.

Fig. 3 shows a system in accordance with the invention comprising a transmitter 31, a receiver 32 and a transmission channel 33, for downloading software from the transmitter to the receiver in accordance with one of the methods described previously, via the transmission channel. During downlink transmission in a broadcasting system, for example, the user equipment would be the receiver and the base station or server the transmitter.

In the case of digital receivers of a broadcasting system, there are currently 2 standards (DVB and DSS) that can be currently supported by the same hardware but, for software size reason, are supported by dedicated software (BSL and application). The invention enables to launch a product with a technology (e.g. compatible with the DSS standard) and to keep the possibility to download in the future a complete different technology (e.g. compatible with the DVB standard). The provider of the receiver comprising software, which is initially compatible with the current system technology, does not need to know the specifications of new systems when launching the product.

The drawings and their description hereinbefore illustrate rather than limit the invention. It will be evident that there are numerous alternatives, which fall within the scope of the appended claims. In this respect, the following closing remarks are made. There are numerous ways of implementing functions by means of items of hardware or software, or both. In this respect, the drawings are very diagrammatic, each representing only one possible embodiment of the invention. Thus, although a drawing shows different functions as different blocks, this by no means excludes that a single item of hardware or software carries out several functions. Nor does it exclude that an assembly of items of hardware or software, or both carries out a function.

Any reference sign in a claim should not be construed as limiting the claim. Use of the verb "to comprise" and its conjugations does not exclude the presence of elements or steps other than those stated in a claim. Use of the article "a" or "an" preceding an element or step does not exclude the presence of a plurality of such elements or steps.

Claims:

1. In a transmission system, method of downloading software programs into a storage unit, the software programs including a boot code and an application code, the boot code allowing downloading the application code, the storage unit comprising at least a current boot code, the method comprising:
- upon a download request, downloading a new boot code in a location, which does not overwrites the current boot code,
- indicating that the new boot code replaces the current boot code,
- downloading a new application code associated to the new boot code in a location, which does not overwrites the new boot code,
- indicating that the new application code is valid.

2. In a transmission system, method of downloading software programs into a storage unit, the software programs including a boot code and an application code, the boot code allowing downloading the application code, the storage unit comprising at least a current software program stored including a current boot code stored in the storage unit at a first position, the method comprising:
- defining a boot sector for jumping on a position of the storage unit where a boot code is stored in order to validate the use of said boot code, the boot sector initially pointing on the first position, where the current boot code is stored,
- upon a download request, downloading a new software program in a second position including a new boot code and a new application code,
- jumping on the second position where the new boot code is stored.

3. Method as claimed in claim 2, wherein the step of jumping on the second position where the new boot code is stored is followed by :
- replacing the current boot code with the new boot code at the first position,
- jumping on the first position.

4. Method as claimed in claim 2, wherein the boot sector is located in a protected storage area of the storage unit.

5. Method as claimed in claim 2, wherein the boot sector is located in a protected storage area separate from the storage unit.

6. Method as claimed in claim 2, wherein the current boot code is stored in a protected area of the storage unit, which area can be unprotected to be overwritten under specific software conditions.

7. Method as claimed in claim 2, wherein the new software program is stored in an area of the storage unit, which area can be protected and unprotected to be overwritten under specific software conditions.

8. Method as claimed in claim 2, wherein the new software program includes an intermediate application code, which is a link between the current application code and the new application code enabling a user to parameterize the new software program.

9. Receiver for receiving software programs transmitted by a transmission system, the receiver comprising means for carrying out the method as claimed in any of the claims 1 to 8.

10. Receiver as claimed in claim 9, wherein said means for carrying out the method as claimed in any of the claims 1 to 8 include a file system.

11. Receiver as claimed in claim 9, wherein the storage unit is a persistent memory enabling to protect / unprotect memory area upon software instructions.

12. Transmission system comprising a transmitter for transmitting software programs and at least a receiver for receiving said software programs, the receiver comprising means enabling to carry out the method as claimed in any of the claims 1 to 8.

13. Computer program product for a receiver computing a set of instructions, which when loaded into the receiver, causes the receiver to carry out the method as claimed in any of the claims 1 to 8.

14. Signal for carrying a computer program, the computer program being arranged to carry out the method as claimed in claim 1.

Abstract.

The invention relates to a method of downloading software programs into a storage unit. The software programs include a boot code and an application code, the boot code allowing downloading the application code. The storage unit comprises at least a current boot code and a current application code. The method comprises:

- upon a download request, downloading a new boot code in a location, which does not overwrites the current boot code,
- indicating that the new boot code replaces the current boot code,
- downloading a new application code associated to the new boot code in a location, which does not overwrites the new boot code,
- indicating that the new application code is valid.

Fig. 1

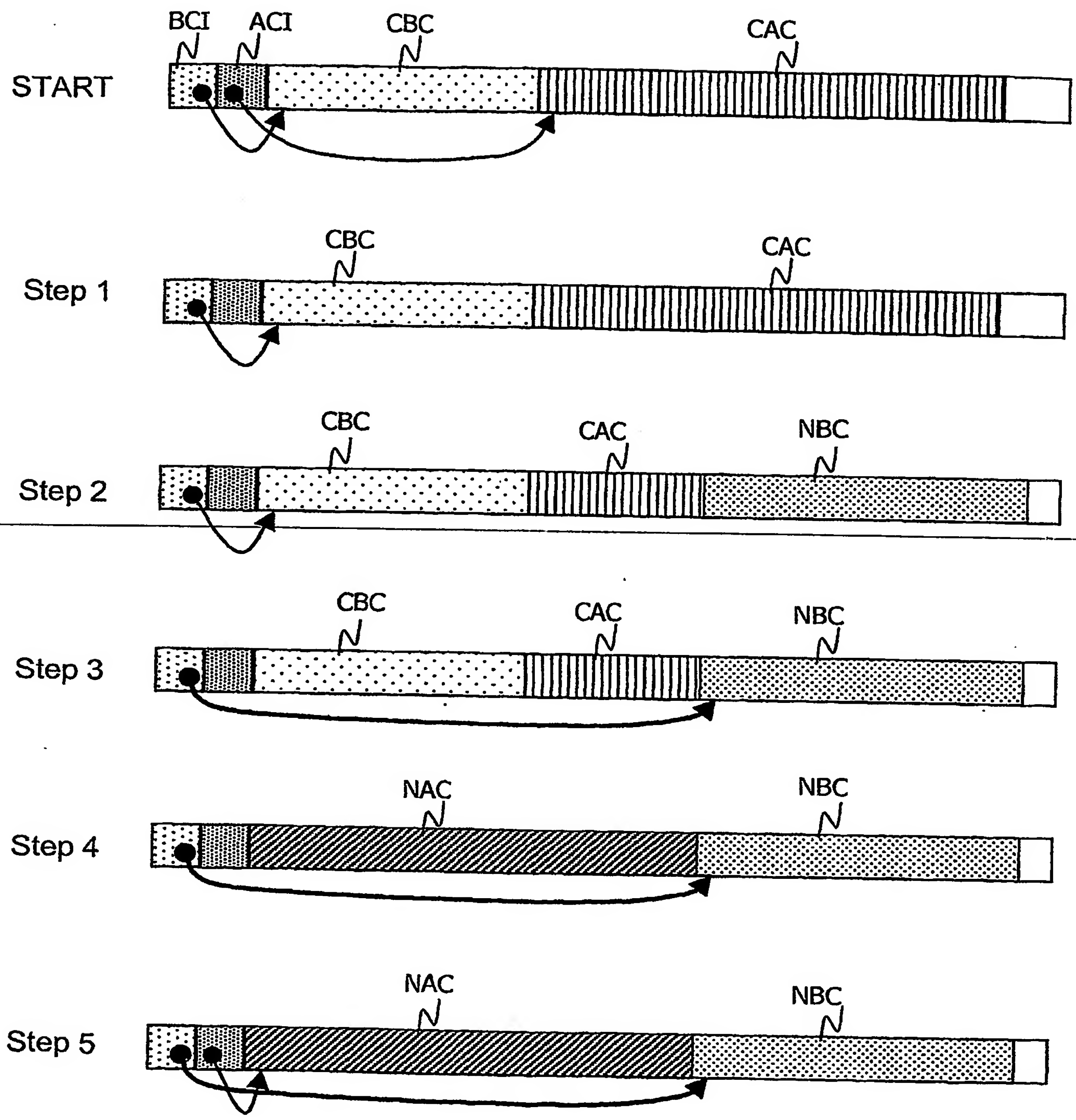
START
BCI
ACI
CBC
CAC
Step 1
CBC
CAC
Step 2
CBC
CAC
NBC
Step 3
CBC
CAC
NBC
Step 4
NAC
NBC
Step 5
NAC
NBC

FIG. 1

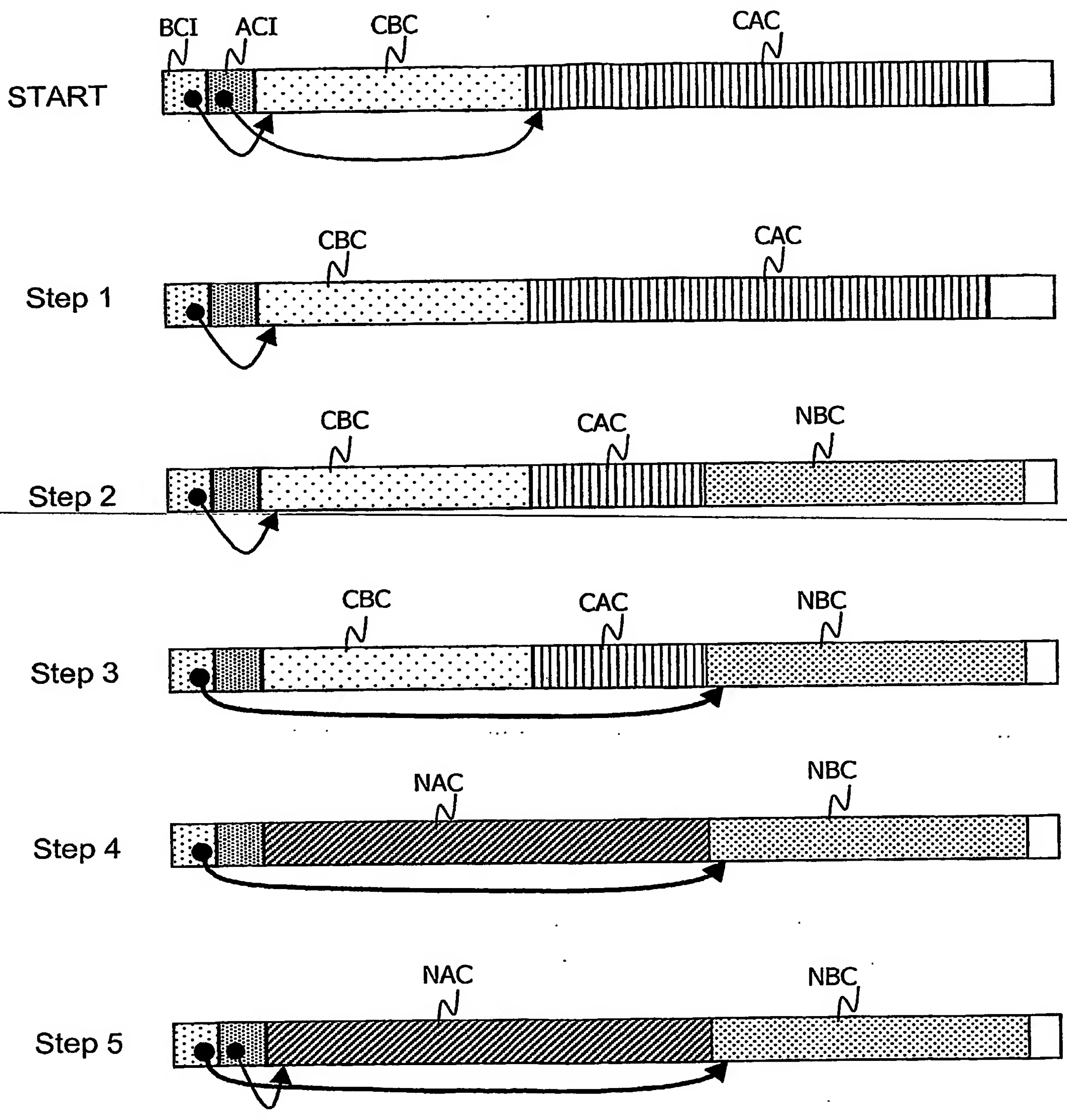
BCI
ACI
CBC
CAC
START
Step 1
CBC
CAC
Step 2
CBC
CAC
NBC
Step 3
CBC
CAC
NBC
Step 4
NAC
NBC
Step 5
NAC
NBC

FIG. 1

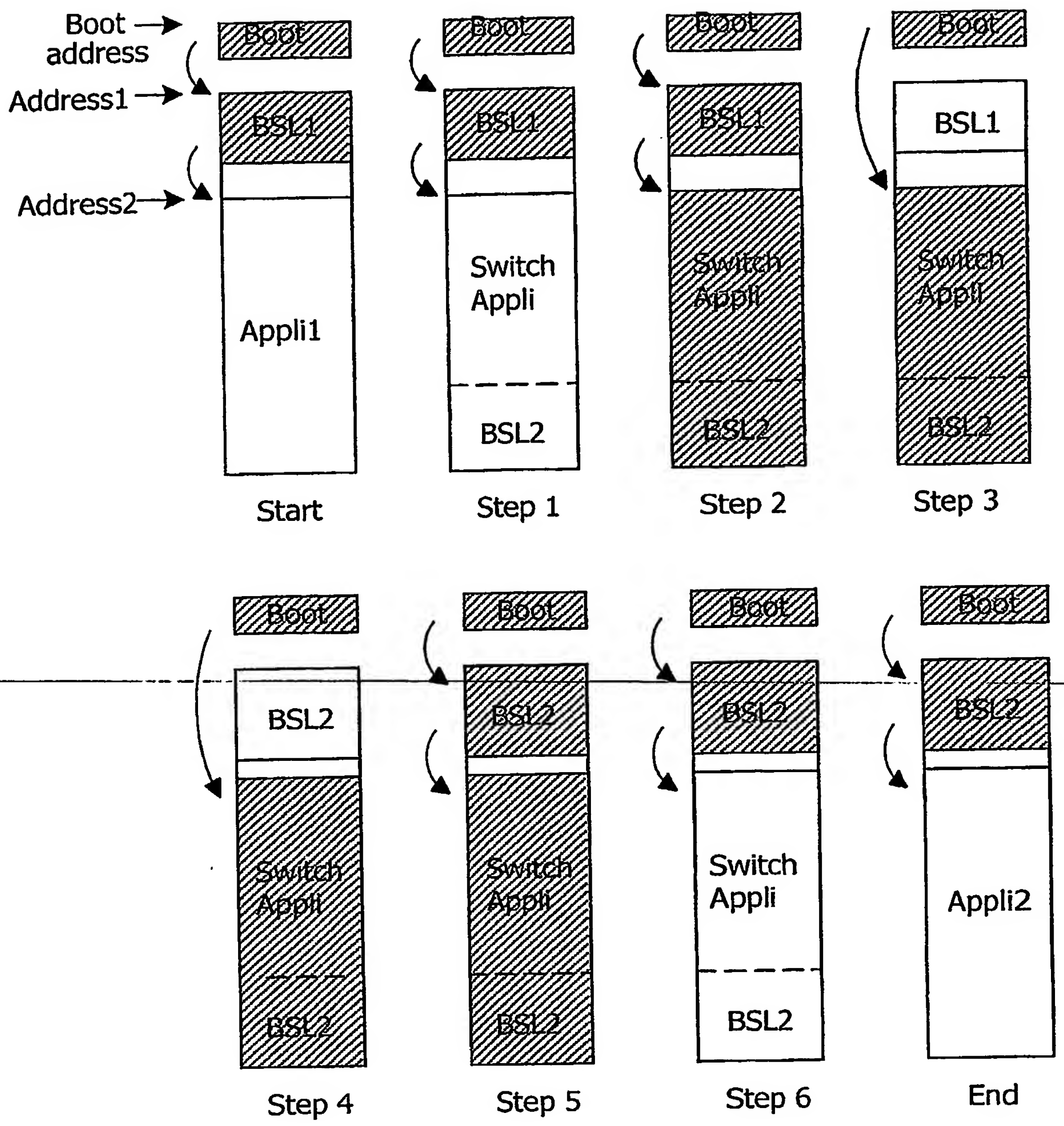
Boot
address
Address1
Address2
Boot
BSL1
Appli1
Start
Boot
BSL1
Switch
Appli
BSL2
Step 1
Boot
BSL1
Switch
Appli
BSL2
Step 2
Boot
BSL1
Switch
Appli
BSL2
Step 3
Boot
BSL2
Switch
Appli
BSL2
Step 4
Boot
BSL2
Switch
Appli
BSL2
Step 5
Boot
BSL2
Switch
Appli
BSL2
Step 6
Boot
BSL2
Appli2
End

FIG. 2

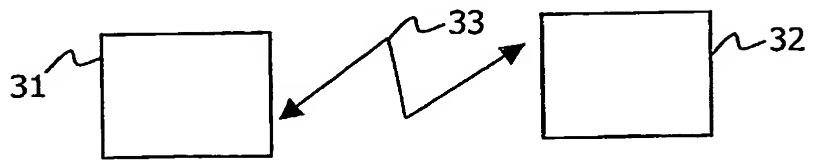
31
33
32

FIG. 3